X-REEact: Fighting Runtime Variances across Time and Space

Bruce Childers[†], Jack Davidson[*], Mary Jane Irwin[+], Mahmut Kandemir[+], Mary Lou Soffa[*]
[†]University of Pittsburgh, [*]University of Virginia, [+]Pennsylvania State University
Contact author: Bruce Childers, childers@cs.pitt.edu

Emerging CMPs (chip multiprocessors) are not only taking over the desktop and enterprise computing market but they are also positioning themselves as building blocks of future exascale systems due to the exceptional computing efficiency they offer in a single chip.

An inherent characteristic of CMPs that presents a significant obstacle is runtime variation: energy consumption, thermal behavior, and process variation will vary across identically-designed components of a CMP, producing a negative impact on application power consumption, reliability and performance. Runtime variation has been identified as one of the key problems that could block further scaling of circuits if not properly addressed. It not only causes reliability problems itself, but it also makes power-saving techniques more susceptible to errors. Furthermore, diverse applications react to runtime variations differently. For example, while a class of HPC applications have self-healing properties which help them cope with variations in a natural fashion, other classes of HPC applications can experience substantial error rates under high runtime variance. Clearly, this observation rules out any one-size-fits-all type of solution. For example, conventional circuit/microarchitecture level solutions built upon the worst-case assumptions will not fly in an exascale regime since they would cause too much power at runtime and this power consumption would not be justified for many applications. Instead, what is needed is a low cost, low footprint, scalable and adaptive solution that can be reconfigured at runtime based on the dynamic needs of threads, applications, and workloads.

Motivated by this pressing need, we propose X-REEact, a novel lightweight and scalable application-level runtime system with the following unique characteristics:

1. An X-REEact instance is initiated at runtime along with application threads, and its main task is the effective adaptation of application threads to runtime variations in most effective manner. The precise objective of an X-REEact instance is determined at runtime based on the dynamic characteristics of the application to which it is attached. For example, if the application is cache intensive, the X-REEact instance will change memory layout of data and/or reorder computations so that critical data accesses do not experience high latencies and cumulative cache access latencies of different threads are balanced.

2. An X-REEact instance will be aware of architectural heterogeneity, in particular, the "big-core, small-core" dichotomy. Using this information, it will be able to make the best decisions for the corresponding application.

3. An X-REEact instance will be "morphable" at runtime in terms of the resources it exercises and metrics it targets. For example, if it detects that temperature is becoming a problem in the current phase of application execution, it will spawn helper threads whose job is to migrate select threads from hot cores to cold cores.

4. X-REEact instances can form "alliances" among themselves if the involved HPC applications belong to the same workflow (e.g., they have producer-consumer relationship).

5. An X-REEact instance will talk to the OS "on behalf of the application". For example, it will ask resources from the OS, let the OS know the dynamic requirements of its application, etc.

6. X-REEact will be built with "scalability as the first-class parameter". Scalability will be achieved by 1) minimizing overheads, 2) carefully (but quickly) weighing pros and cons of potential optimizations before applying them, and 3) minimizing resource footprint (to save power).

Related Work

Virtualization for HPC systems is related to the run-time system approach taken by X-REEact. The Palacios virtual machine monitor does *full-system virtualization (FSV)* across multiple nodes [6,7]. It allows different guest OSes to be run on a host executive, such as Kitten on a Cray XT4 [7]. Palacios applies FSV techniques to improve I/O pass through, workload paging, and controlled preemption [1,6]. In contrast, X-REEact does *application-level virtualization (ALV)* to monitor, control, and transform application execution, rather than providing FSV capabilities [5, 12]. X-REEact uses application-specific transformations, such as dynamic rethreading [9] and cache contention mitigation [5, 13], to adapt execution to runtime variations. These techniques require application information and mediation that are unavailable at the full-system level.

Other related work includes Nesbit and Smith's virtual private machines to manage spatial and temporal resources in CMPs [10]. The Xen hypervisor has been extended to support application-specific resource management [11]. Cuvillo described a thread virtual machine for applications to achieve full resource utilization. Noll et al. describe a virtual machine to let programmers hide the heterogeneity of the Cell processor architecture [4]. Other related work includes Multikernel [2] and Log-based architecture [3]. While these systems provide application-specific management capabilities, X-REEact differs in it targets multiple nodes and adapts (transforms) applications to mitigate runtime variation.

Assessment

- Challenges addressed: Fault avoidance, containment and enhanced detection; OS/R survivability; Power management under dynamically changing resources; Runtime/application management of memory; Management and optimization for design and runtime heterogeneity; Scalability and parallelization of the runtime system in a range of system scales.
- Maturity: We designed and implemented a prototype of X-REEact for a single CMP node. The prototype is flexible and extensible to support different management policies and optimizations [5]. We demonstrated X-REEact in handling thermal emergencies [5], dynamic workload partitioning [8,9], thread mapping to mitigate resource contention [13]. This work established X-REEact's feasibility and promise. We are now developing reliability optimizations, methods to scale to multiple nodes, distributed algorithms for self organization, and application–system–X-REEact interfaces.
- Uniqueness: We are not aware of any existing runtime system that can morph itself (to serve the application in the most efficient way) as well as help the application morph itself to cope with runtime variations.
- Novelty: At any given time, our system can have multiple X-REEact instances, each serving to a different application, trying to maximize a different objective, and cooperating with the OS on behalf of its application. This facility will also make the OS design for future exascale machines easier by shifting functionality to X-REEact instances.
- Applicability: X-REEact is expressly intended to accommodate a range of systems and runtime management and optimization strategies. It will provide the interfaces and support to permit custom approaches, as demonstrated in our initial prototype. As such, X-REEact has broad applicability beyond high-performance computing to general-purpose systems as well (e.g., thermal/energy management for desktop computers).
- Effort: With the foundation from our current prototype, we anticipate it will take one year to scale the approach to multiple nodes, including developing algorithms and interfaces for X-REEact's dynamic hierarchical organization. To incorporate reliability, energy and performance optimizations on multiple nodes, we anticipate another one and a half years of work. To refine and finalize X-REEact's implementation, we anticipate it will take a a final six months.

References

[1] C. Bae, J. Lange, and P. Dinda. Enhancing virtualized application performance through dynamic adaptive paging mode selection. In *8th International Conference on Autonomic Computing (ICAC)*, 2011.

[2] A. Baumann, P. Barham, P.-E. Dagand, T. Harris, R. Isaacs, S. Peter, T. Roscoe, A. Schupbach, and A. Singhania. The multikernel: a new OS architecture for scalable multicore systems. In *ACM SIGOPS 22nd Symposium on Operating Systems Principles*, 2009.

[3] S. Chen, B. Falsa, P. B. Gibbons, M. Kozuch, T. C. Mowry, R. Teodorescu, A. Ailamaki, L. Fix, G. R. Ganger, B. Lin, and S. W. Schlosser. Log-based architectures for general-purpose monitoring of deployed code. In *1st Workshop on Architectural and System Support for Improving Software Dependability*, 2006.

[4] Juan del Cuvillo. *Breaking away from the OS Shadow: A Program Execution Model Aware Thread Virtual Machine for Multicore Architectures*. PhD thesis, University of Delaware, Newark, DE, USA, 2008. Adviser- Guang R. Gao.

[5] Tanima Dey, Wei Wang, Ryan Moore, Mahmut Aktasoglu, Bruce R. Childers, Jack W. Davidson, Mary Jane Irwin, Mahmut Kandemir, and Mary Lou Soffa. C-VEM: A customizable virtual execution manager for multicore platforms. In *Int'l. Conf. on Virtual Execution Environments*, 2012.

[6] J. Lange, K. Pedretti, P. Dinda, P. Bridges, C. Bae, P. Soltero, and A. Merritt. Minimal overhead virtualization of a large scale supercomputer. In *Proceedings of the 2011 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2011)*, 2011.

[7] J. Lange, K. Pedretti, T. Hudson, P. Dinda, Z. Cui, L. Xia, P. Bridges, M. Levenhagen, R. Brightwell, A. Gocke, and S. Jaconette. Palacios: A new open source virtual machine monitor for scalable high performance computing. In *Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2010)*, 2010.

[8] Ryan Moore and Bruce R. Childers. Inflation and deflation of self-adaptive applications. In *6th Int'l. Symp. on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2011.

[9] Ryan Moore and Bruce R. Childers. Using utility prediction models to dynamically choose program thread counts. In *IEEE Int'l. Symp. on Performance Analysis of Systems and Software (ISPASS)*, 2012.

[10] Kyle J. Nesbit, James Laudon, and James E. Smith. Virtual private machines: A resource abstraction. Technical report, In University of Wisconsin - Madison, ECE TR, 2007.

[11] D. Nikolopoulos, G. Back, J. Tripathi, and M. Curtis-Maury. VT-ASOS: Holistic system software customization for many cores. In *Int'l. Symp. on Parallel and Distributed Processing (ISPDP)*, pages 1–5, 2008.

[12] Kevin Scott, Naveen Kumar, Silva Velusamy, Bruce Childers, Jack Davidson, and Mary Lou Soffa. Retargetable and reconfigurable software dynamic translation. In *International Symposium on Code Generation and Optimization*, pages 36–47, 2003.

[13] Wei Wang, Tanima Dey, Jason Mars, Lingjia Tang, Jack W. Davidson, and Mary Lou Soffa. Performance analysis of thread mappings with a holistic view of the hardware resources. In *IEEE Int'l. Symp. on Performance Analysis of Systems and Software (ISPASS)*, 2012.